The K-Nearest Neighbors Algorithm

Beginner Track - Introduction to ML Workshop #2

Attendance code: **Weather**
Linktree: bit.ly/btrack-w22-feedback
Discord: bit.ly/ACMdiscord

acm acm.ai


acm
at UCLA
Intern Applications
Due October 29 | Week 5
If you are interested in being an ACM Officer, apply to our Internship Program!
No experience required

A Motivating Example

Suppose you see this creature in the park...

Cat or dog?

Some terms

- **Feature** - some **property** of the object that impacts the **target**

  Eg. for an animal, the sharpness of its claws is a feature.

- **Target**/**Label** - the **true** value of what we are trying to predict.

  Eg: In our current example, the target would be whether the creature is a cat or a dog.

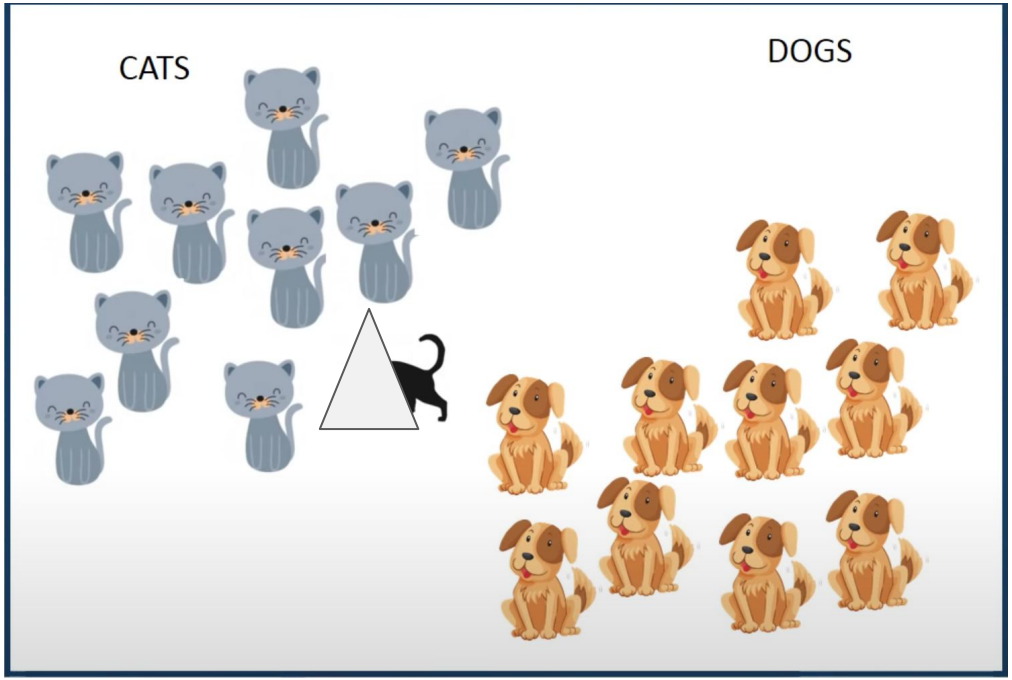
Feature Differences between Cats and Dogs

- Sharp claws (used for climbing)
- Shorter ears
- Meows

- Dull claws
- Longer ears
- Barks

Is it a Cat or a Dog?

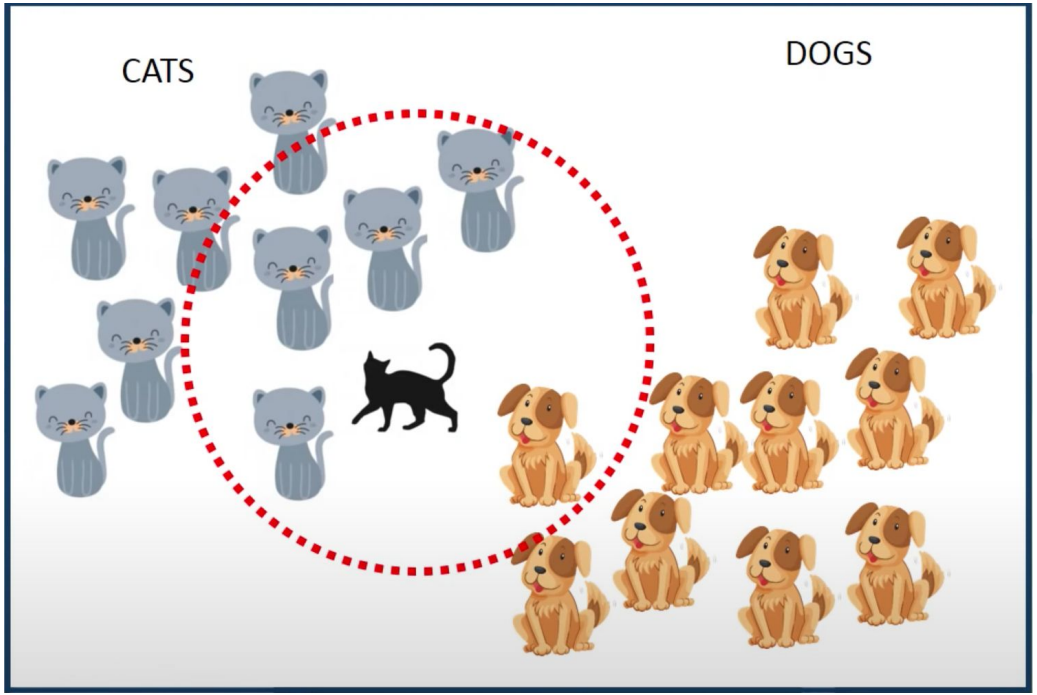
Sharpness of claws



Length of ears

It is, in fact, a Cat!

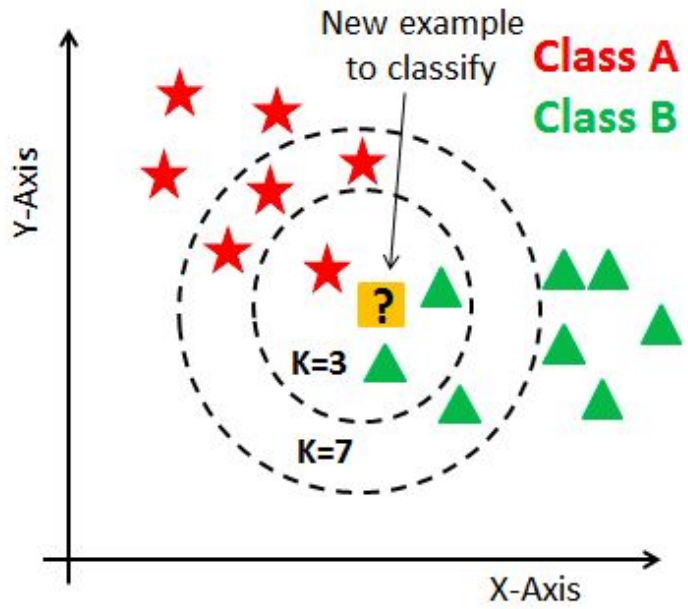
Sharpness of claws



Length of ears

K-Nearest Neighbors in a Nutshell

- Goal: classify the new data point based on how its *neighbors* are classified
- Observe that the outputs are **categorical** in our cats vs dogs example



Let’s Formalize What We Discussed

A Sample Dataset

Index	X	Y
0	3.1	4.0
1	51.0	9.2
2	6.5	3.5
3	89.0	7.9

How would you determine what a neighbor is?

How would you determine what a neighbor is?

- Our data-points live in 2 dimensional space
- We can compute **distances** using the Euclidean distance measure
- So for our points (x_1, y_1) and (x_2, y_2)

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

We know what ***neighbors*** are, what's the ***nearest*** part?

The **nearest** neighbors to a given point ***p*** are the ones which have the **minimal Euclidean distances** to the that point

What does the **K** in KNNs represent?

K is just a **hyperparameter** (a value that *we* choose) to determine **how many** nearest neighbors to look at

The algorithm to find the K nearest neighbors

- Let's say we choose one point **p** = (X, Y)
- Compute the distance between **p** and every other point in our data set using the Euclidean formula
- Choose the *K* data-points that are the closest to **p**
- We can then classify **p** using those K nearest points

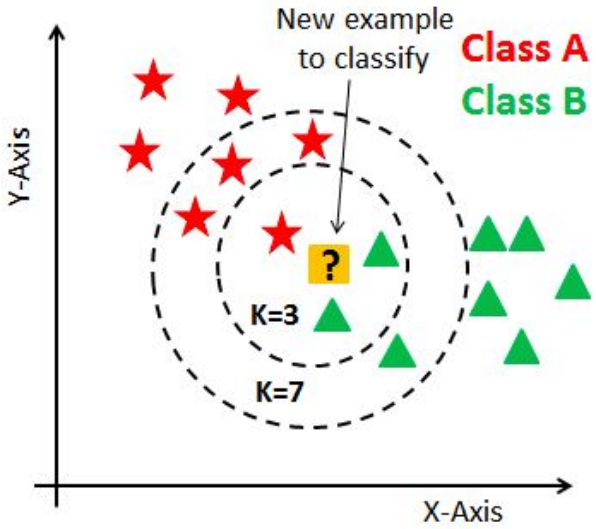
Now that we have the K nearest points

- We want to **classify** our new point **p**
- We look at the **classes** of those K nearest points
- Determine the class that is the **most common**
  - i.e. the *mode*
- We can predict that **p** belongs to this class!

Putting it all together

Given a dataset of points and a new point **p** to classify:

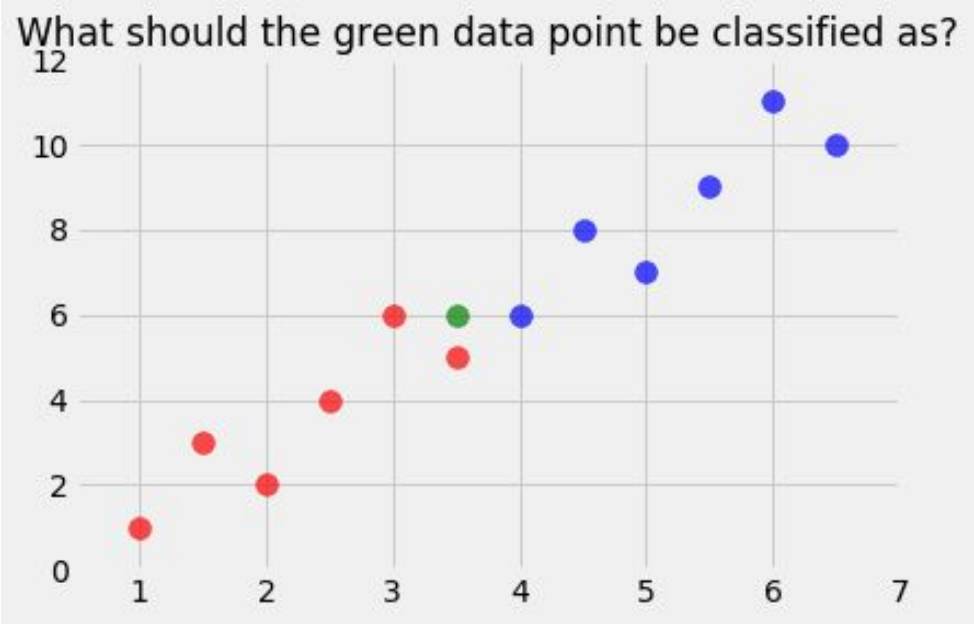
1) Choose a value for **K**
2) Compute the distances between **p** and every point in the dataset.
3) Using these distances, determine the K points that are closest to **p**
4) Look at the classes of these K points and choose the class that is the most common
5) This is the class that **p** belongs to!



Quick Quiz

What should the green data point be classified as? Choose k = 3.

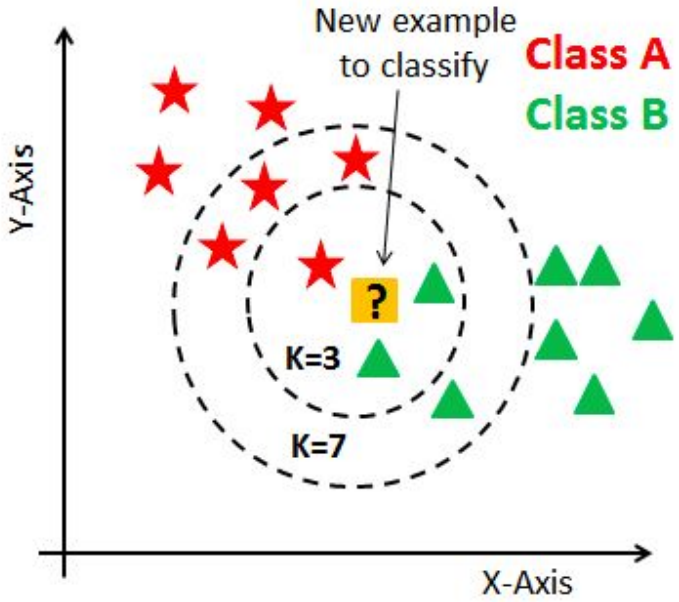
a. Red
b. Blue
c. Green
d. None of the above



The Hyperparameter ‘K’

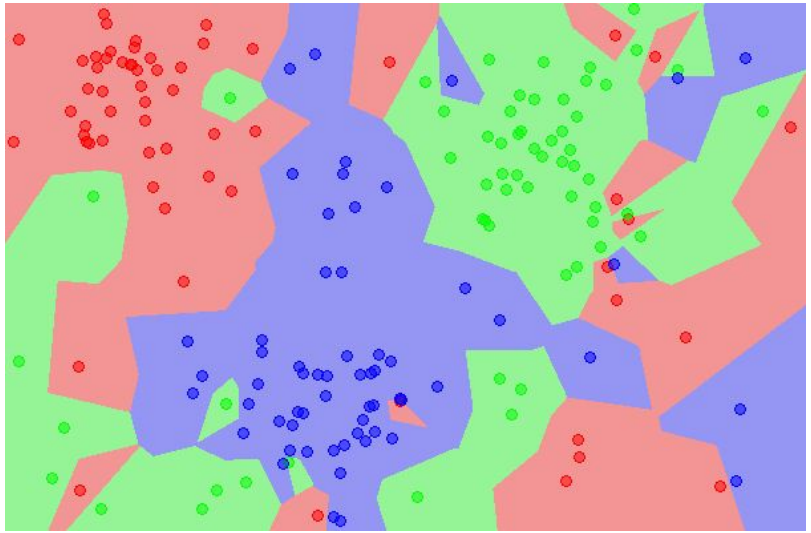
How do we choose the factor ‘K’?

- Choosing the right value of k is important for better accuracy
- Here’s an example of a dilemma
  - At k = 3, we classify ? as a triangle
  - But at k = 7, we classify ? as a star
- So which one is correct?
  - We should try both and see which gives us a better accuracy

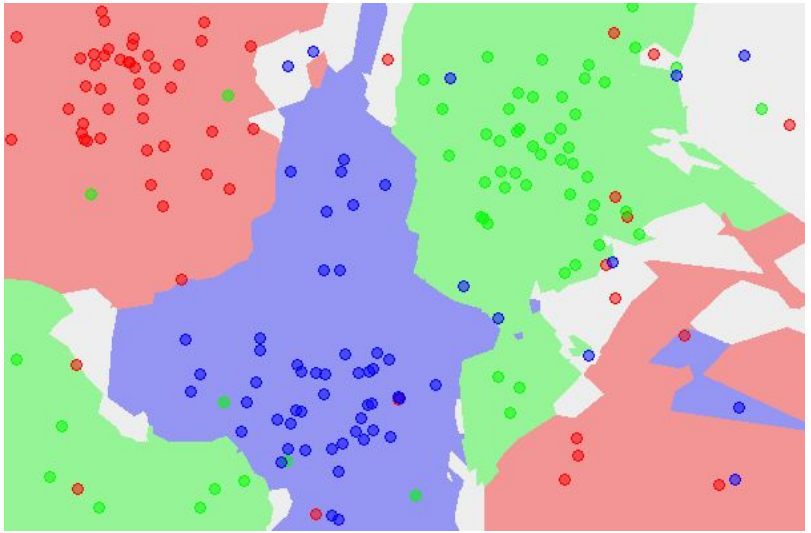


How do we choose the factor ‘K’?

Choosing K = 1



Choosing K = 5



How do we choose the factor ‘K’?

- What would happen if we pick a k that is too low?
  - Think about noise/outliers in the dataset
- What would happen if we pick a k that is too high?
  - As an extreme, think about if k = size of dataset
- A general rule of thumb for choosing k
  - k = sqrt(n), where n is the total number of data points
  - **Odd** value of k to avoid a potential tie in the voting process
  - This is a good *starting point* for k. You should try different values to see what gives the best result.

Quick Quiz

If there are 170 data points in your KNN algorithm, what k value should you pick?

a. 11
b. 12
c. 13
d. 14

Where does the *training* come in?

- The point of training is to give the model information from our dataset
- In the case of of a KNN we can just use the dataset directly
- So there's no explicit training process
- That's why KNN is called a **lazy** learning algorithm as opposed to an **eager** learning algorithm

Training and Testing Algorithm

Can you identify any potential pitfalls of KNN?

```
class KNearestNeighbor(object):

  def __init__(self):
    pass

  def train(self, X, y):
    self.X_train = X
    self.y_train = y

  def test(self, x_test, k=1):

    dists = np.linalg.norm(self.X_train.T - x_test, axis=1).T
    sortedIdxs = np.argsort(dists)
    closest_y = self.y_train[sortedIdxs[:k]]
    y_pred = np.argmax(np.bincount(closest_y));

    return y_pred
```

Pros and Cons

- Pros:
  - No training required!
  - Simple to implement
  - Works well for small datasets
- Cons:
  - Does not scale well to large datasets
  - Curse of dimensionality - computationally and in representations

Case Study

How does KNN work in practice?

- Consider a dataset having 2 features: **X** and **Y**
- Each sample is labeled as **0** or **1**

X	Y	Label
51	167	0
62	182	1
69	176	1
64	173	1
65	172	1
56	174	0
58	169	1
57	173	1
55	170	0

How does KNN work in practice?

57	170	?

- Given this train dataset, we have to classify this new point **p** as either in class 0 or 1 using KNN
- To find nearest neighbors, we will use Euclidean distance

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

How does KNN work in practice?

- Here, the last column shows the distance between each training sample and **p**
- Total number of data points = 9
- **k** = sqrt(9) = 3
- To find the K-Nearest neighbors to **p**, we need to sort our table by each sample's distance to **p**

X	Y	Label	Distance
58	169	1	1.4
55	170	0	2
57	173	1	3
56	174	0	4.1
51	167	0	6.7
64	173	1	7.6
65	172	1	8.2
62	182	1	13
69	176	1	13.4

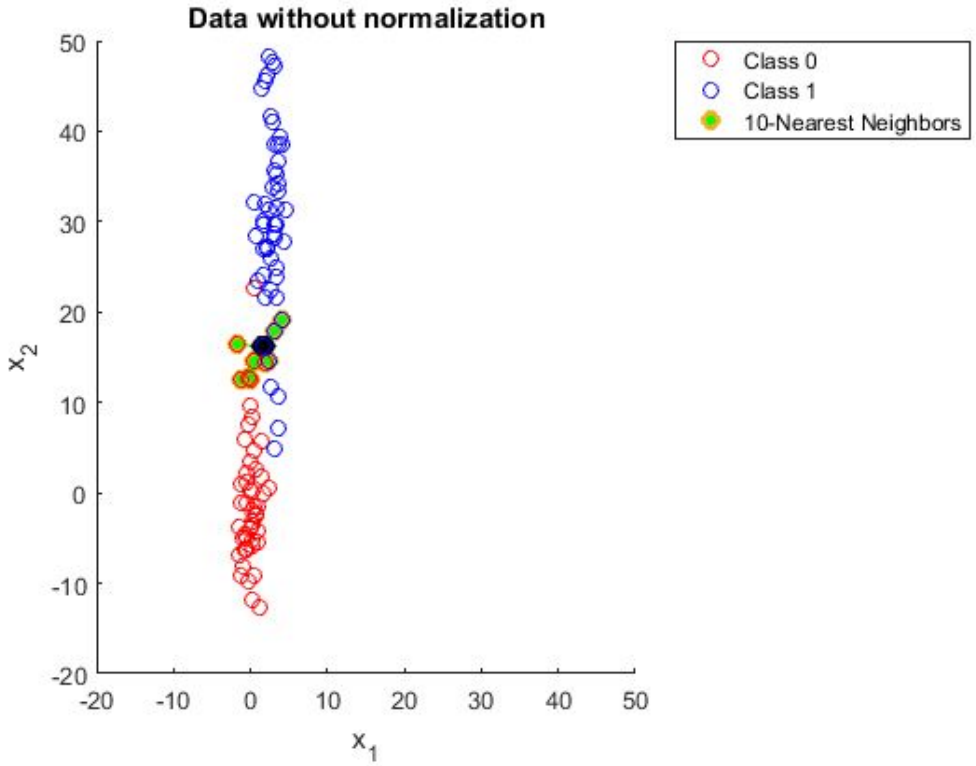
Recap of KNN

- Given a dataset and a new point to classify
- Compute the distance between **p** and each sample in the dataset
- Sort the dataset by each sample's distance to **p**
- Select **K** entries from our dataset closest **p**
- Find the most common classification (mode) of these **K** entries
- This is the classification we give to the test data point **p**

The Importance of Data Normalization for KNNs

What is the issue with this dataset?

Specifically, what is it about this dataset that might make our KNN model perform suboptimally?

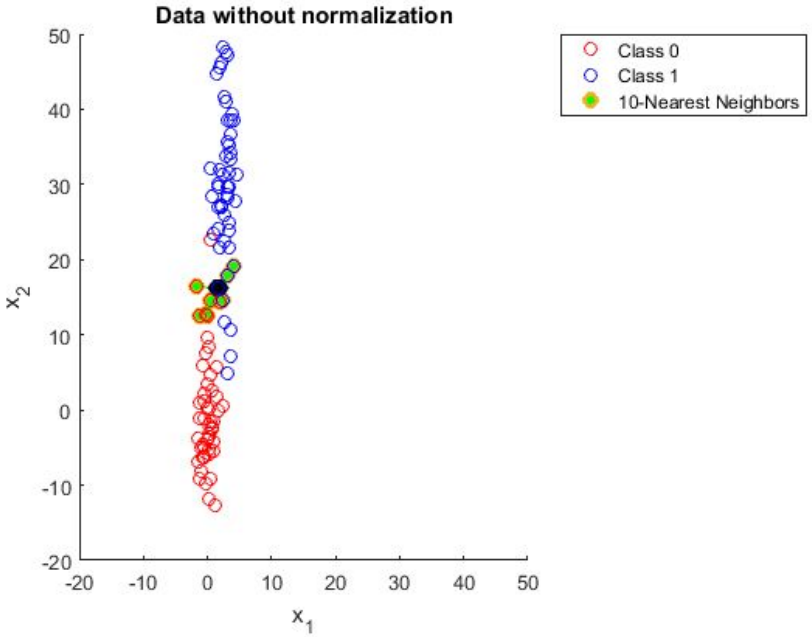


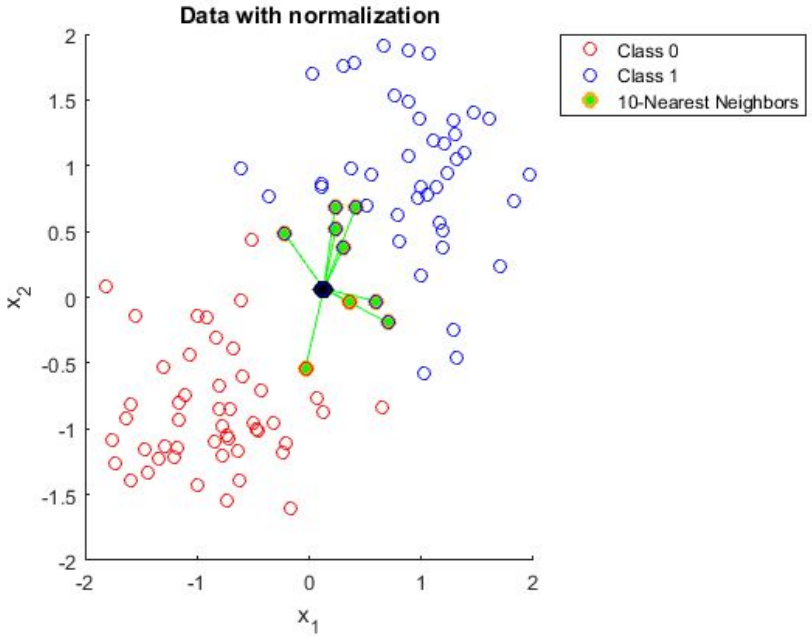
What do we mean by ‘normalization’?

- f *is one of our features (a column)*
- μ is the mean value in that column
- σ is the standard deviation of that column

$$f_{normalized} = \frac{f - \mu}{\sigma}$$

Normalize the Data Beforehand!





Code Demo

Coding up a KNN

- Use the link on the slide/feedback form to access a copy of the notebook so you follow along if you'd like!

tinyurl.com/btrack-w22-w2-demo

Thank you! We'll see you next week!

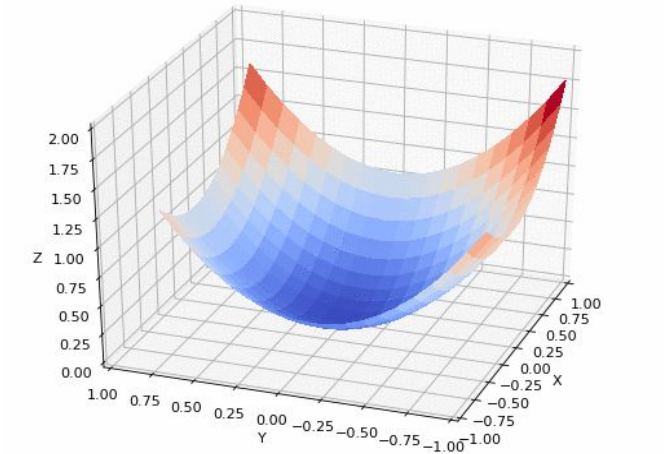
Please fill out our feedback form: bit.ly/btrack-w22-feedback

Next week: Linear Regression

If you liked y = mx+b, you're gonna love this

Today's event code: **Weather**

FB group: facebook.com/groups/uclaacmai



acm acm.ai